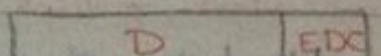
Link layer :- - nodes → hosts and routers
- links → Communication channels between nodes (wired / wireless / LANs
- frame encapsulates datagram (Coming from network layer)
- responsible for transfer datagram from physically node to another over a link

Link layer Services :-
- flow Control :- pacing between adjacent sending and recieving nodes
- error detection : reciever detects presence of errors (Signals sender for retransimission or dropped frames
- error Correction :- reciever identifies and Carect bits error
- half-duplex and Full-duplex :- with half-duplex nodes at both ends of link can transmit but not at same time
- Link layer is implemented :- in each host, in adaptor aKa NIC (Network interface card or on achip

- Adaptors Communication :-
- Sending side : encapsulates datagram in frame, adds error checking bits, rdt, flow control ...
- recieving side : Looks for error, rdt, ..., extract datagram, passes to upper layer

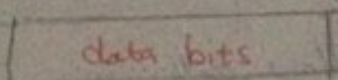
- Error detection :-
- EDC : error detection and Correction bits



- not 100% reliable : protocol may miss some errors rarely, larger EDC field yields better detection and correction

- parity checking :-

Single bit parity :-
detect single bit errors



two dimensional bit parity
detect and corrects single bit errors

- Internet checksum : used in transport layer only

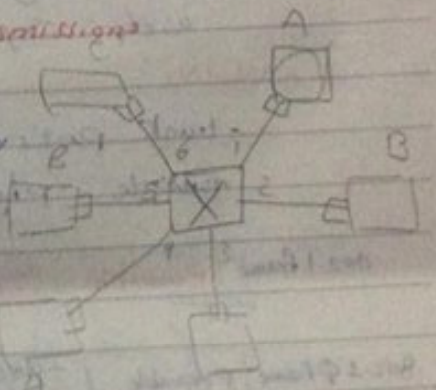
Ethernet:

- Common MAC protocol and frame format
- diff speed
- different physical layer media: fiber, cable

- Ethernet Switch: (link layer device)

- Store, forward ethernet frames - examine incoming frame's MAC address, selectively forward frame, use CSMA/CD to access segment
- transparent: hosts doesn't know it's there
- Plug and play, self learning (don't need configuration)

- Switch: multiple simultaneous transmissions



- hosts have dedicated direct connect to switch, switch buffer packets
- Ethernet protocol used but no collision (full duplex

- Switch forwarding table

- each switch has a routing table
- each entery (MAC add of host, interface to reach host, time stamp)
- Switch learns which hosts can be reached through which interface (sender بيبعت بيانات الـ frame بيستقبل Switch الـ)

Slide 65

- Interconnecting switches

self-learning they connect exactly the same as in single switch case

Switches Vs Routers

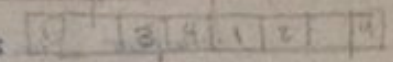
- both are store and forward and have forwarding table

Switch: link layer device (examine link. L header)
- Learn forwarding table using flooding, learning, MAC add

routers: network layer device
- Compute tables using routing Algorithms, IP add

- Channel partitioning MAC protocols:

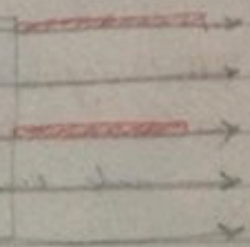
1- TDMA: time division multiple access



- access to channel in rounds, each station get fixed length slot in each round, unused slots go idle

2- FDMA: - frequency

- Channel spectrum divided into freq. bands each station assigned fixed band, unused transmition time in bands go idle.



- Random access protocols:

- when a node has packet to send, it transmits at full channel rate R, there is no coordination between nodes and if 2 or more transmitting nodes collision happens
- they specify how to detect collision and how to recover from it

1- Slotted ALOHA

assumption: all frames same size, time devided into equal slots, nodes start to transmit only slot bigining, nodes synchronized and if 2 or more nodes transmit in slot, all nodes detect collision

operation: when node obtains fresh frame, transmits in next slot

if no collision: node can send new frame in next slot

if collision: node retransmits frame in each subsequent slot

pros:

- Single active node can continuously use full rate of channel
- highly decentralized
- Simple

Cons:

- Collision, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

- Efficiency: Long-run fraction of successful slots

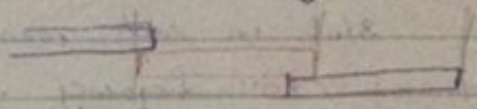
prop that given node has success in a slot $= P(1-P)^{N-1}$

prop that any node $= NP(1-P)^{N-1}$

Max Efficiency $= NP^*(1-P^*)^{N-1}$

P^* probability that maximize efficiency

- Pure (unslotted ALOHA)
  - Simpler, no synchronization
  - when frame arrives it's transmitted immediately
  - collision probability increase



- Efficiency:

P(success by given node) = P(node transmits) · P(no other node transmits in $[t_0-1, t_0]$) · P(no other node transmits in $[t_0-1, t_0]$)

$= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1} = p(1-p)^{2(N-1)}$

even worse than slotted ALOHA

(CSMA) Carrier Sense Multiple Access: Listen before transmit (human analogy): don't interrupt other
- If channel sensed idle → transmit entire frame
  busy → defer transmission
- Collisions: can still occur
  - entire packet transmission time wasted (distance and propagation delay play role in collision probability

* CSMA/CD: as in CSMA, carrier sensing, deferral
  - collision detected within short time, colliding transmissions aborted, reducing channel wastage
  - Collision detection: easy in wired LANs: measure signal strength, compare transmitted, recieved signals
  - difficult in wireless LANs: recieved signal strength overwhelmed by local transmission strength
  - Ethernet CSMA/CD algorithm Slide 33
  - Efficiency:

T_{prop} = 2 prop delay between 2 nodes

T_{trans} = time to transmit max-size frame

$$\text{Efficiency} = \frac{1}{1 + 5 t_{prop} / t_{trans}}$$

- better than ALOHA (simple, cheap, decentralized)

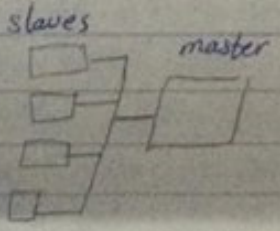
- taking turns MAC Protocol:-

Channel Partitioning
: share channel efficiently and fairly at high load
inefficient at Low Load : delay in channel access, 1/N bandwidth allocated even if only 1 active node

random access
- efficient at Low load : 1 node can fully utilize channel
- high load : collision overhead

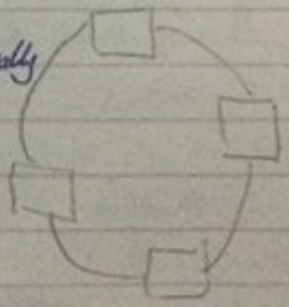
taking turns
look for the best of both worlds

1- Polling



- master node invites dumb slaves nodes to transmit in turn
- Concerns : Single point of failure (master)
  Latency - polling overhead

2- token passing :



- Control token passed among nodes sequentially
- Concerns: token overhead - latency
  Single point of failure (token)

address resolution protocol

MAC Address and ARP

- 32 bit IP address → network layer (like postal address)
- (MAC / LAN / physical / Ethernet) address (frame) (like social security No.)
  - used locally to get frame from interface to another physically connected interface
  - 48 bit address - burnt onto NIC ROM

LAN address

- each adapter on LAN has unique LAN address

ARP protocol: Same LAN

B لـ dg حتى يعرف A لـ

- B's MAC address not in A's ARP table
- A broadcasts ARP query packet يحتوي B IP address
  - ARP query لـ جميع nodes لـ في
  - His MAC address لـ A في query لـ يستقبل B
  - unicast A MAC address لـ حتى frame لـ
  - ARP table لـ (IP-to-MAC) B في pair لـ save في A

- ARP protocol is "plug and play" net administrator لـ لا حاجة لـ arp table يكون nodes لـ

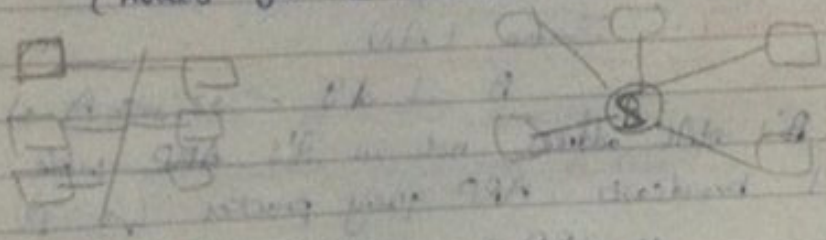
Addressing: routing to another LAN

- A create IP datagram with IP source A, dest B
- A creates link-layer frame with R's (switch/router) Mac address as dest (sent from A to R)
- frame sent to R datagram removed passed up to IP B
- R creats frame to B's MAC address as dest

rdt reliable data transfer

Ethernet : dominant wired LAN technology
- Cheap - widely used LAN technology
- Simpler, cheaper than LANs: ATM

physical topology:
bus : all nodes in same collision domain (can collide with each other)
today star : active switch in center
Separate Ethernet protocol for nodes to collide with each other



Frame Structure:
ethernet frame ... encapsulation IP data gram ... Sending adapter

Preamble	dest add	Src add	type	data (payload)	CRC

Preamble used to: synchronize reciever, sender clock rate
10101010 pattern ... 7 bytes ... byte ... 10101011

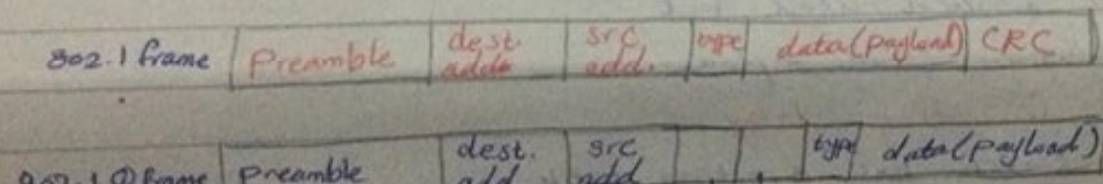
address: 6 byte src, dest MAC address
broadcast add ... matching dest. add ... frame ... adapter
network layer ... data
higher layer protocol

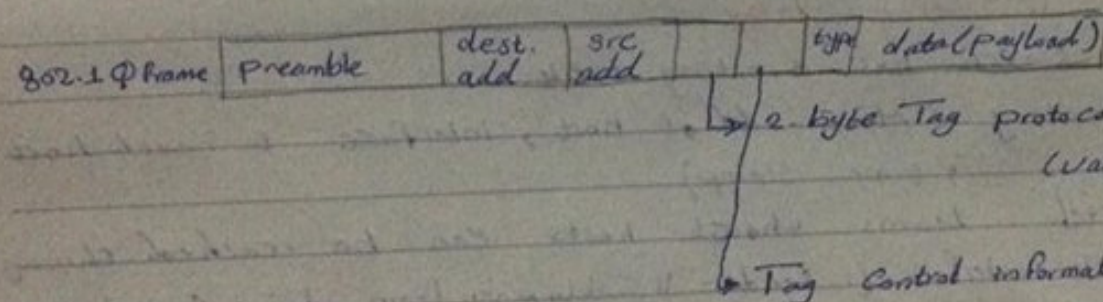
type:
CRC: error detected

Ethernet:
- Connectionless (no Handshaking)
- unreliable: reciever NIC doesn't send acks or nacks to sending NIC
sender ... receiver ... data dropped
higher layer rdt (TCP)
- Unslotted CSMA/CD with binary back off

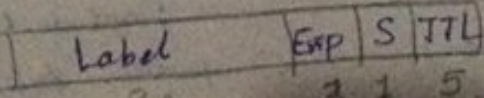
VLANs: switches supporting VLAN capabilities can be configured to define multiple virtual LANs over single physical LAN infrastructure.

* port-based VLAN: switch ports grouped so that single physical switch operates as multiple virtual switches
  - traffic isolation
  - dynamic membership
  - forwarding between VLANs done via routing (as with separate switches)
* VLANs spanning multiple switches: slide 74
  - trunk port: carries frames between VLANs defined over multiple physical switches





MPLS (Multiprotocol Lable Switching)
initial goal: high-speed IP forwarding using fixed length lable instead of IP address
- fast look up using fixed length identifier (rather than shortest prefix matching) - borrowing ideas from VC



4/5

- MPLS ~~capte~~ capable routers aka Label-Switched routers (forward packets to outgoing interface based only on Label value don't inspect IP address

MPLS vs IP paths Slide (79, 80)

- IP paths routing: path to destination determined by dest. address alone
- MPLS routing: path can be based on Source and dest. add
  - fast reroute: precompute backup routes in case of link failure

- Data Center networks

10's to 100's of thousand of hosts often closely coupled in close proximity: - e-business (Amazon)
- Content-Servers (youtube, Microsoft) - Search engines, data mining (google

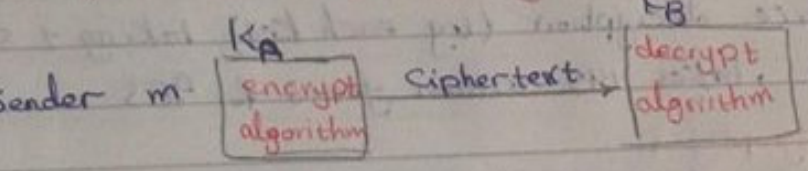
- Challenges: - multiple applications each serving massive no. of clients, managing / balancing load, avoiding processing, networking, data bottlenecks
- Load balancer: - app layer routing
  - receives ~~data~~ external client requests
  - directs workload within data center
  - returns results to external client slide 85

Ch 8 : Network security

- What is (network) security?
  1- Confidentiality : only sender and reciever should understand message contents (Sender encrypts msg, reciever decrypts it)
  2- authentication : Sender and reciever want to confirm identity of each other
  3- message integrity : Sender and reciever want to ensure message not altered (in transit or afterwards) without detection
  4- access and availability : Services must be accessible and available to users

- What can bad guys do?
  - eavesdrop : intercept messages
  - actively insert messages into connection
  - impersonation : can spoof source address in packet
  - hijacking : take over ongoing connection by removing sender or reciever, inserting himself in place
  - Denial of service : prevent service from being used by others (eg. overloading resources)

- principles of cryptography:



m → plaintext msg K_A, K_B → encryption, decryption keys
$K_A(m)$ → ciphertext $m = K_B(K_A(m))$

- Breaking an encryption scheme:
  - ciphertext only attack : bad guys have ciphertext the can analyze - two approaches (brute force search through all keys - statistical analysis)
  - know-plaintext attack : they've plaintext corresponding to ciphertext
  - chosen-plaintext attack : they can get ciphertext for chosen plaintext

* Symmetric key cryptography :-

Sender and reciever share same key (K_S) (secret key)

(Simple) [1] Substitution cipher : Substituing one thing for another

- monoalphabetic cipher : letter
- encrypt. key : mapping from set of 26 letters to set of 26 letters

[2] n-substitution ciphers : $M_1, M_2 \dots, M_n$

Cycling pattern : eg. $n=4$ M_1, M_2, M_3, M_4, M_1

- encrypt. key : n-substitution ciphers and cyclic pattern key neednot be just n-bit pattern

[3] DES : Data encryption standard

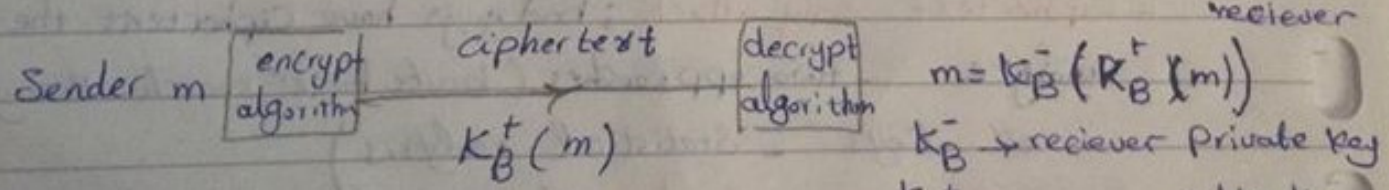
- 56-bit symmetric key, 64-bit plaintext input
- block cipher with cipher block chaining
- How secure? 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
- making it more secure : 3DES (encrypt 3 times with 3 different keys Slide 15

[4] AES : advanced Encryption Standard

- processes data in 128 bit blocks
- 128, 192 or 256 bit keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

* Public key cryptography :

- Sender reciever don't share secret key
- public encryption key known to all
- private decryption key known only to reciever



given K_B^+ public key, it should be impossible to compute private key K_B^- Slide 20

5

RSA : Rivest, Shamir, Adelson algorithm

- message : just a bit pattern (can be represented by an integer number. So encrypting a message is equivalent to encrypting a number)
- ex: m = 1001000 → decimal 145 (corresponding no.
- to encrypt m, we encrypt the corresponding no. which gives new no. (ciphertext)

- Creating public/private key pair
  1- Choose 2 large prime number p, q (e.g 1024 bit)
  2- Compute $n = pq$, $z = (p-1)(q-1)$
  3- Choose e ($e<n$) that has no common factors with z. (e, z relatively prime)
  4- Choose d such that ed-1 exactly divisible by z (ed mod z = 1)
  5- public key is (n, e) , private key (n, d)

- encryption, decryption
  1- given (n, e) and (n, d)
  2- to encrypt message m $c = m^e \bmod n$
  3- to decrypt recieved bit pattern c
  $m = c^d \bmod n$
  So $m = \underbrace{(m^e \bmod n)}_{c}{}^d \bmod n$

- why does RSA work

$c^d \bmod n = m$ where $c = m^e \bmod n$

fact For any x, y : $x^y \bmod n = x^{(y \bmod z)} \bmod n$

thus $c^d \bmod n = (m^e \bmod n)^d \bmod n$

$= m^{ed} \bmod n$

$= m^{(ed \bmod z)} \bmod n$

$= m^1 \bmod n$

$= m$

- another important property:

$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$ why?

$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$

$= m^{de} \bmod n$

$= (m^d \bmod n)^e \bmod n$

- Why RSA, secure?

Suppose you know public key (n, e) How hard is to determind d? you need to find factors of n without knowing p, q (really hard)

- RSA in practice: session keys

- DES is at least 100 times faster than RSA
- So we use ~~pup~~ public key crypto to establish secure connection then, stablish second key - symmetric key session key - for encrypting data (Sender and reciever use RSA to exchang a symmetric key, once they have it they start symmetric key cryptography

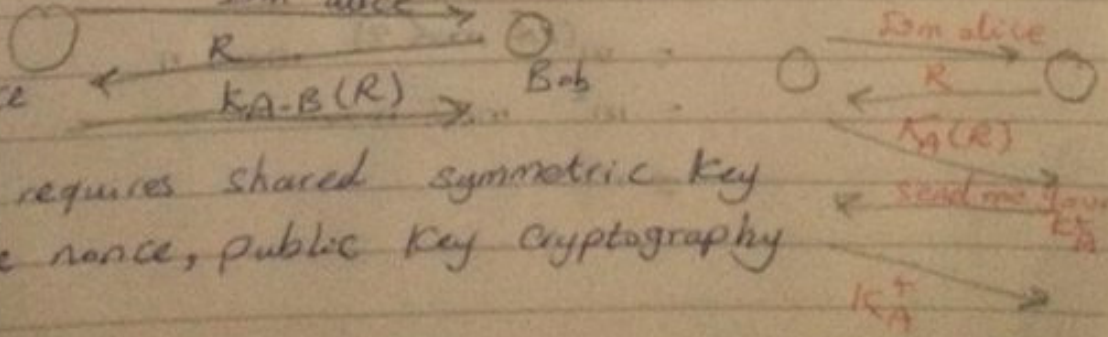
* Authentication:

- playback attack: hijacker records sender's packets and later plays it back to recievor

- Goal: avoid playback attack

ap4.0 - nonce: number (R) used once-in-a-lifetime

- to prove (confirm) sender identity to reciever reciever ~~sender~~ sends R (nonce) to sender and Sender must return it "encrypted with shared secret key



- this requires shared symmetric key

ap5.0 use nonce, public key cryptography

apps of security hole

- man in the middle attack : hijacker poses as sender to reciever and as reciever to sender
- difficult to detect cause every side recieves everything

Digital Signature : cryptographic technique analogous to hand-written signature (sender digitally signs document) verifible, nonforgeable

- Simple digital signature for msg m:
  - Sender signs msg m with his private key K_B^-
  - → non-repudiation

Message Digests

goals: fixed-length, easy-to-compute digital "finger print" by applying H hash fn to m and get fixed size message digest H(m)

Hash fn properties :-

- many-to-1
- produces fixed-size msg digest (finger print)
- given message digest x, computationally infeasable to find m such that x

Internet Checksums poor crypto hash fn

It has some of hash fn properties like :- many-to-1

- produces fixed length digest of message (16-bit sum)

But : given m with given hash value, its easy to find another message with same hash value

Digital Signature, Signed message digest

graph slide 49

Hash fn Algorithms

- MD5 hash fn widely used (computes 128-bit msg digest in 4-step process - can't construct m out of x)
- SHA-1 is also used (160-bit message digest

* Certification authority (CA) : binds public key to particular entity E (person, router)
  - E registers its public key with CA
    - E must provide "proof of identity" to CA
    - CA creates certificate binding E to its public key
    - Certificate signed by CA
  - to get someone's public key → apply CA's public key to his certificate, you get his public key

- Securing e-mails
  - to send confidential email
    Sender 1- Sender generates random symmetric private key, K_s
    2- encrypts messages with K_s
    3- also encrypts K_s with reciever's public key K_B
    4- Sends both $K_s(m)$ and $K_B(K_s)$ to reciever
    reciever : 1- uses his private key to decrypt and recover K_s
    2- Uses K_s to decrypt $K_s(m)$ to get m
  - to provide Sender authentication msg integrity
    - Sender digitally signs message
    - sends both msg (in the clear) and digital signature
  - to provide secrecy, Sender authentication, message integrity
    Sender uses 3 keys: his own private key, reciever's public key, newly created symmetric key

---

* Securing TCP Connections : SSL (secure Socket layer)
  - widely deployed security protocol (Supported by allmost all browsers, webserver - http)
  - mechanism : Woo 1994 implementation : Netescape
  - Variation TLS (transport layer security
  - provides (Confidentiality, integrity, authentication)
  - Original goals: web e-commerce transactions encryption especially Credit card no. - Web server authentication

8)

- optional client authentication - minimum hassle in doing business with new merchant)
- available to all TCP applications

SSL and TCP/IP

application
TCP
IP

normal app

Applications
SSL
TCP
IP

app with SSL

- SSL provides app programming interface (API) to apps
- C and Java SSL libraries/classes readily available

Toy SSL: a simple Secure channel

- handshake: Sender and reciever use their certificates, private keys to authentication each other and exchange shared secret
- Key derivation: Sender and reciever use shared secret to derive set of keys
- data transfer: data to be transfered is broken up into series of records
- Connection closure: special msgs to securely close connection

* Toy: Simple handShake Slide 65

* Toy: Key derivation:

It's bad to use same key for more than one crypto operation

- 4 keys:

K_C, K_S → encrypt key for data sent from client to server and vice versa

M_C, M_S → MAC key for data sent

- keys derived from KDF (Key derivation fn)

* Toy: data records

- Why not encrypt data in constant stream as we write it to TCP? - where would we put the MAC?

If stand: no message integrity until all data processed

- instead: break stream in series of records - each record carries a MAC and as it arrives reciever can act on it
- issue: reciever need to distinguish MAC from data (use variable-length record) | Length | Data | MAC |

* Key: Sequence numbers
  - Problem: attacker can capture or replay or reorder records
    Solution: put sequence no. into MAC (M_K, Sequence # data)
  - Problem: attacker could replay all records
    Sol: use nonce

* Key: Control information
  - problem: truncation attack (attacker forges TCP connection close segment - one or both sides think there is less data than there is)
  - Solution: - records type (type 0 for data, type 1 for closure
    - MAC = MAC(M_K, Sequence # type # data)

| length | type | data | MAC |

- SSL Cipher Suit: (Public-key algorithm - symmetric encryption algorithm - MAC algorithm)
  - SSL supports several Cipher Suits
  - negotiation: client and server agree on cipher suit (Client offers choice - Server picks one

Real SSL: - handshaking (I)

1- Server Authentication 2- negotiation
3- Establish keys 4- Client authentication (optional)

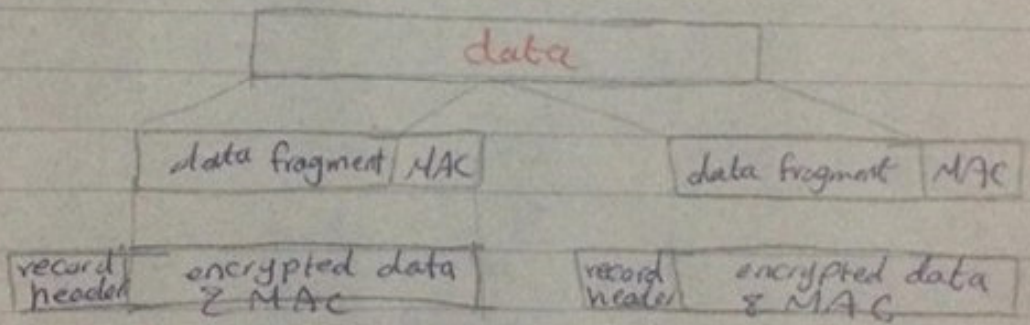
- Handshaking (II)

1- Client send list of algorithms it supports with client nonce
2- Server choose algorithm from list; sends back (choice + Certificate + Server nonce)
3- Client verifies certificate; extract server's public key; generates pre-master secret, encrypts with Server's public key; sends to Server

4- Client and server compute encryption and MAC keys (from 3)
5- Client sends a MAC of all the handshake messages
6- Server

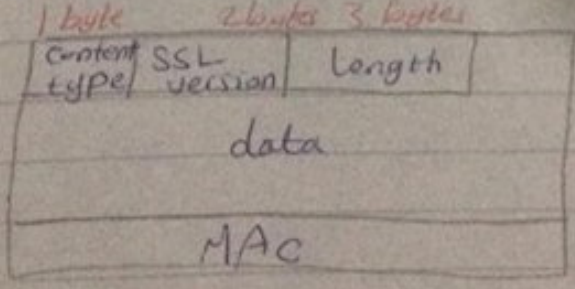
- Handshaking (3) Last 2 steps protect handshake from tamper
  - Client tepically offers range of algorithms (strong & weak)
  - man in the middle could delete strong algorithms from list
  - last 2 steps prevent this (last 2 msgs are encrypted)
- Hand Shaking (4) ex slide 76
  - We use 2 different random nonce for each connection to make encryption keys different

- SSL record protocol



record header: Content type, version, length
MAC : includes sequence no., MAC key M_x
fragment, each SSL fragment 2^{14} bytes (16 Kbyte)

→ SSL record format:



ex: slide 79

Key derivation:

- (client nonce, server nonce, pre-master secret input) → Psudo random no. generator → produces master secret

- (master secret, new nonces) → random-no. generator → key block

- key block sliced and diced
  - client, server MAC keys
  - client encryption
  - initialization vectors (IVs)
  - *